1. Tell us the differences between uncontrolled and controlled components.

Controlled components refer to components that are directly controlled by the application meaning the application manages the state and behavior of the component. These components are typically custom-built for the specific application and have a clear interface with the application. Examples of controlled components include buttons, text inputs, and custom UI elements.

Uncontrolled components, on the other hand, are components that manage their own state internally and do not have a clear interface with the application. These components are often pre-built and provided by third-party libraries or frameworks. Examples of uncontrolled components include HTML form elements such as checkboxes and radio buttons.

2. How to validate React props using PropTypes?

React, PropTypes is a library that is used to validate the props of a component. PropTypes is a built-in feature of React, which provides a way to specify the type and structure of props that a component expects to receive. By using PropTypes, developers can ensure that the components are being used correctly and avoid bugs caused by incorrect usage of props.

Here is some common PropTypes validation:

- PropTypes.number
- PropTypes.bool
- PropTypes.func
- PropTypes.string
- PropTypes.object
- PropTypes.array

3. Tell us the difference between nodejs and express js.

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It allows developers to write server-side JavaScript code that can be executed on the server. Node.js provides a non-blocking I/O model that allows for fast and scalable server-side applications. With Node.js, developers can build back-end services, APIs, and other server-side applications.

Express.js, on the other hand, is a web framework built on top of Node.js. It provides a simple and flexible API for building web applications and APIs. Express.js provides features like routing, middleware, and templating that make it easier to build web applications. With Express.js, developers can build web applications and APIs quickly and easily.

4. What is a custom hook, and why will you create a custom hook?

In React, a custom hook is a JavaScript function that uses one or more built-in React hooks to encapsulate a certain piece of logic and make it reusable across different components. Custom hooks allow developers to extract and reuse stateful logic from components, making the code more organized, readable, and maintainable. Custom hooks allow developers to abstract away complex logic and functionality from components, reducing the amount of code duplication and making it easier to maintain the codebase. By encapsulating common logic in a custom hook, developers can avoid code duplication and improve the overall code quality. Custom hooks are prefixed with the word "use" to indicate that they are built on top of the built-in React hooks. A custom hook can use any of the built-in React hooks, including useState, useEffect, useContext, and useRef.